REMARKS

Claims 1, 3-12, 14-48 are pending in the present application. Reconsideration of the claims is respectfully requested.

I.   **35 U.S.C. § 103, Alleged Obviousness, Claims 1, 3-4, 7-12, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 46-48**

The Office Action rejects claims 1, 3-4, 7-12, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 46-48 under 35 U.S.C. § 103(a) as being allegedly obvious over Ng et al. (U.S. Patent No. 6,385,618) in view of Sarkar (U.S. Patent No. 6,418,448). Because this rejection is essentially the same as in the previous Office Action, this rejection is respectfully traversed for the reasons stated in the previous Response filed July 16, 2003, the remarks of which are hereby incorporated by reference. The following remarks are provided in rebuttal of the Examiner's statements in the present Office Action beginning on page 3, paragraph 2 of the Final Office Action.

In the July 16, 2003 Response, Applicant argued that the Ng reference does not teach or suggest, with respect to claim 1, which is representative of the other rejected independent claims 12 and 46, determining a delete action based on the structure of the relational database, generating database modification commands based on the determined delete action, or sending the database modification commands to a relational database server, where the relational database server deletes the object data from the relational database based on the database modification commands. In response, the Examiner, on page 3 of the present Office Action states the following:

> Ng does teach the relational database and object relational database schema that is a logical structure of a database from which the users including database administrator (e.g. to change the structure f database or database schema) and end-user (e.g. to update, add or delete database records) enable to customize the database by using manipulation database operations or database modifications (col. 1, lines 60-67, col. 2, lines 12-53). Database modifications are including insertion, update deletion...ect. to the structure of the database or internal data structure (known as database data structure) (col. 4, lines 14-38). All database modifications are performed via an executable programmed source code for

customizations structure of database or database schema. The codes are read on or examined on or queried the database schema that constructs a database data structure to reflect the schema and the customizations of modifications are reflected in the source code or a program file, for example such as a database administrator wants to delete a column in a database table or table (col. 5, lines 23-67 and col. 6, lines 1-31). Thus Ng does teach the deletion or modifications the structure of a database via a programmed source code file and the arguments of applicants are not persuasive.

Applicant respectfully disagrees that Ng teaches or suggests determining a delete action based on the structure of the relational database, generating database modification commands based on the determined delete action and sending the database modification commands to a relational database server, where the relational database server deletes the object data from the relational database based on the database modification commands. As stated in the July 16, 2003 Response, Ng only teaches updating an object model and then generating a database data structure based on this object model. Nowhere is it taught or suggested to determine a delete action based on a structure of the database.

Even though Applicant contends that Ng does not teach determining a delete action based on the structure of the relational database, Ng does teach that updates made to the database by the administrator are incorporated into the source code at column 7, lines 5-12, which reads as follows:

> FIG. 5 depicts a flowchart of the states performed by the object-relational mapping tool when mapping a database while maintaining the programmers customizations to the object model as reflected by the source code. In other words, the database has been updated by the database administrator, as shown in FIG. 6, and the programmer now wants to incorporate these updates into the source code without losing his customizations.

While this section may teach generating source code based on an updated data structure of the relational database, Applicant respectfully submits that Ng does not teach determining a delete action based on the structure of the relational database or generating database modification commands based on the determined delete action. The Final Office Action alleges that this feature is taught at column 1, lines 60-67 and column 2, lines 12-53, those sections read as follows:

Object-relational mapping tools have been created to facilitate development of application programs that utilize a relational database. A relational database stores data in tables having rows (records) and columns (fields). The tables are usually interrelated, and thus, there is a logical structure imposed on the database. This logical structure is known as a schema. Each table may have a primary key, comprising one or more columns that uniquely identify a row.

(Column 1, lines 60-67)

Object-relational mapping tools read database schema information and automatically generate source code from the database. This source code contains a number of classes whose interrelationships reflect the logical structure, or schema, of the database. A class, such as a Java.TM. class, is a data structure containing both data members that store data and function members (or methods) that act upon the data. The source code may contain one class for each table in the database, and each class may have a data member for each column in the corresponding table. Additionally, the classes contain function members that are used to both get and set the values for the data members and, eventually, update the database.

By using an object-relational mapping tool, a programmer can automatically generate source code to facilitate their database application development. After the source code is generated, the programmer writes code to interact with only the classes in the source code and not the database, thus hiding the complexities of interacting with the database from the programmer. This allows a programmer who is familiar with object-oriented programming to code against familiar classes and not unfamiliar, sometimes cumbersome to use, database query languages.

Although beneficial to programmers, conventional object-relational mapping tools suffer from a limitation. When a programmer runs the object-relational mapping tool, it generates source code with classes that reflect the structure of the database at that time. However, during the lifetime of the database, it is common for a database administrator to change the schema of the database (e.g., add a new field or table). Likewise, it is common for the programmer to update the classes in the source code (e.g., change a field name or delete a field). As such, both the classes and the database tend to evolve and change over time. Conventional object-relational mapping tools, however, are of little help in such situations. These tools can only remap the database to generate classes that contain the database modifications, but which do not contain the programmer's modifications. Therefore, the programmer's changes are lost and must be manually recreated, thus wasting significant development time. It is therefore desirable to improve object-relational mapping tools. (emphasis added)

(Column 2, lines 12-35)

In column 1, lines 60-67, Ng teaches that databases store data in tables and the
tables are interrelated in logical structures known as a schema. In column 2, lines 12-35,
Ng teaches object-relational mapping tools that read a database schema and create source
codes from the database. Neither of these sections teach or suggest determining a delete
action based on a structure of the database or generating database modification
commands based on the determined delete action. There simply is no mention anywhere
in Ng to determine a delete action based on a structure of the database or to generate
database modification commands based on the determined delete action.

The Final Office Action alleges that these features are taught at column 4, lines
14-38. The cited section is included in the column 4, lines 14-54, which reads as follows:

> Methods and systems consistent with the present invention provide
> an improved object-relational mapping tool that integrates both
> customizations to source code and modifications to a database.
> Accordingly, the programmer does not have to recreate their
> customizations to the source code when the database changes, thus saving
> significant development time over conventional systems.
> Overview
> In accordance with methods and systems consistent with the
> present invention, the improved object-relational mapping tool maps a
> database by first querying the database to determine its schema and then
> by creating an internal data structure (known as the "database data
> structure") representing that schema. From this data structure, the object-
> relational mapping tool creates an object model containing all of the
> information necessary to generate classes and then creates source code
> containing a number of Java classes that may be used by a programmer to
> interface with the database.
> At some point during the lifetime of the object model, the
> programmer may add customizations to the object model (e.g., rename a
> field) that will be reflected in the source code, and the database
> administrator may likewise make a change to the database schema (e.g.,
> add a column). After the database schema has been changed, the
> programmer may wish to update their source code to reflect the schema
> change while maintaining their customizations. To accomplish this goal,
> the object-relational mapping tool, in accordance with methods and
> systems consistent with the present invention, imports the new schema,
> creates a new database data structure, and compares the original (or
> preexisting) database data structure with this newly created one, noting
> any differences. Having noted the differences, the object-relational
> mapping tool has isolated the changes to the schema, and it then
> incorporates these changes into the existing object model and generates
> new source code. As a result, both the programmer's customizations to the

object model (reflected in the old version of the source code) as well as the changes to the schema made by the database administrator are integrated into the new source code, thus saving the programmer significant development time over conventional systems. (emphasis added)

This section of Ng teaches that the programmer can make changes to the object model and the database administrator can make changes to the database schema, both of which can be isolated by the object-relational mapping tool, incorporated into the existing object model and generated into new source code. However, nowhere in this section, or any other section, of Ng is it taught or suggested that this process involves determining a delete action based on the schema or generating database modification commands based on the determined delete action and sending the database modification commands to a relational database server.

Furthermore, the Office Action alleges that generating database modification commands based on the determined delete action and sending the database modification commands to a relational database server are also taught at column 5, lines 23 to column 6, line 31, which reads as follows:

> Object-relational mapping tool 114 reads database 118 to examine its schema, constructs database data structure 115 to reflect this schema, generates an object model 116 based on database data structure 115, and then creates source code 120 based on object model 116. It should be noted that, at the time object model 116 is generated, the object-relational mapping tool allows the programmer to add customizations, and these customizations will be reflected in the source code 120. For example, the programmer may add a new method, rename a field (and use it for a different purpose), change the attributes of a field (e.g., the type or whether it can accept a null value), or override the mapping of a field. When a field mapping is overridden, that field will not appear in the source code.
>
> FIG. 2 depicts a more detailed diagram of an example of database 118, containing a customer table 202 and an order table 204. The customer table 202 includes a customer ID column 206, and a name column 208. The customer ID column 206 serves as the primary key for the customer table 202. The order table 204 includes order ID column 212, date column 214, and customer ID column 216. The order ID column 212 serves as the primary key for the order table 204. Customer ID column 216 is the foreign key to customer ID column 206, meaning customer ID column 216 refers to the customer ID column 206 in one or more rows. As previously stated, database data structure 115 represents the schema of database 118. Object-relational mapping tool 114 creates database data structure 115 by

querying database 118 to identify its schema and by creating the data structure to reflect the schema. This process is known as "importing" the database schema and is described in further detail below. Once created, database data structure 115 appears as shown in FIG. 3 and includes an object 302, reflecting the customer table 202, and an object 304, reflecting the order table 204. Object 302 contains a list 306 of foreign key objects, if any, each containing the name of the foreign key as well as an indication of the columns that comprise the foreign key. Additionally, object 302 contains a list 308 of the indexes in the customer table 202, where each element of the list is an index object containing an indication of the type of index (e.g., primary, non-unique, and unique) and a list of columns that comprise the index. Object 302 also contains a hash table 310, where each entry in the hash table is a column object 312, 314 containing data for a particular field, including its name, type, and length. Object 304 contains similar information, including a list of foreign keys 316, a list of indexes 318, and a hash table 320 with column objects 322-326 for each field or column.

Using the object-relational mapping tool, the programmer may customize the object model. For example, the programmer may rename the name field to SSN and may subsequently use this field to store the customer's social security number, in which case the customer's social security number will be stored in the name field 208 of the database 118. By making such a customization, it is reflected in the object model 116 shown in FIG. 4A. Object model 116, generated by the object-relational mapping tool, contains the programmer's customization (e.g., the name field has been renamed to SSN). Object model 116 contains objects 400 and 401, representing an intermediate form of the information for a class before it is written as source code. Object 400 contains information for the customer table 202, including a list 402 of relationship objects 403, each containing information indicating a relationship (i.e., a foreign key). For example, relationship object 403 links a field in object 400 with a field in object 401 to which it is linked. Additionally, object 400 contains a hash table 404 with an entry 406, 408 for each field in customer table 202, each entry containing the name and type of the field. Similarly, object 401 contains information for order table 204, including a list 410 of relationship objects 411 and a hash table 412 containing entries 414-418 for each field in order table 204. Although methods and systems consistent with the present invention are described with reference to specific data structures, such as hash tables, one skilled in the art will appreciate that other data structures may also be used. (emphasis added)

This section of Ng describes in detail how the source code is created and changed based on the changes made by the programmer customizing the object model and the database administrator changing the database. There is nothing in this section or, as

shown above, any other section of Ng, that teaches determining a delete action based on a structure of the database, generating database modification commands based on the determined delete action, or sending the database modification commands to a relational database server.

The previous Office Action acknowledges that Ng does not teach the limitation of the relational database server deleting the object data from the relational database based on the database modification commands, but that Sarkar allegedly teaches this feature. However, Sarkar does not provide for the deficiencies of Ng. That is, Sarkar does not teach determining a delete action based on a determined structure of a relational database or generating database modification commands based on a determined delete action. Sarkar teaches a method and apparatus for processing markup language specifications for data and metadata used inside multiple related Internet documents to navigate query and manipulate information from a plurality of object relational database over the World Wide Web.

While Sarkar may teach loading Java classes in a database server as alleged by the Office Action, there is no teaching or suggestion in Sarkar regarding determining a delete action based on the structure of a relational database, generating database modification commands based on the determined delete action, or sending the database modification commands to the relational database server. Loading of classes in a database server does not teach or suggest sending database modification commands that are generated based on a delete action determined based on a determined structure of a relational database.

Furthermore Sarkar is directed to solving a completely different problem than that of Ng. Ng is directed to preserving object model customizations while accommodating changes to database schema. Sarkar is directed to a system for navigating, querying and manipulating information from a plurality of object relational databases over the Internet. One of ordinary skill in the art would not have been motivated to combine these two systems at least because they are directed to two disparate fields of technology.

Moreover, neither reference teaches or suggests the desirability of incorporating the subject matter of the other reference. That is, there is not motivation offered in either reference for the alleged combination. The Office Action alleges that the motivation for the combination is to "obtain database server of a object relational database locating of

elements inside component relational schema with Java classes." While Sarkar mentions locators of elements inside component relational schema, there is no teaching or suggestion that this is a reason why the system of Sarkar should be combined with the system of Ng. Moreover, there is no suggestion in Ng of the desirability to be able to locate "elements inside component relational schema with Java classes." To the contrary, the only teaching or suggestion to even attempt the alleged combination is based on a prior knowledge of Applicant's claimed invention thereby constituting impermissible hindsight reconstruction using Applicant's own disclosure as a guide.

Furthermore even if one were somehow motivated to combine Sarkar with Ng, the result would not be the invention as recited in independent claim 1. Since neither reference alone teaches or suggests determining a delete action based on a determined structure of a relational database, generating database modification commands based on a determined delete action, or sending the generated database modification commands to a relational database server, the alleged combination would still not teach or suggest these features.

These arguments with regard to Sarkar and the combination of Sarkar and Ng were originally presented in the Response filed July 16, 2003. The Final Office Action fails to address these arguments. Therefore, the Final Office Action has not shown where Applicant's arguments are in error or provided any reasons as to why Applicant's arguments are not to be considered persuasive. Thus, Applicant respectfully submits that these arguments are valid and should be considered by the Examiner.

Independent claims 12 and 46 recite similar features to those of claim 1 and thus, are allowable over the alleged combination of Ng and Sarkar for similar reasons. That is, claim 12 recites that the data processor determines a delete action based on the structure of the relational database, generates database modification commands based on the determined delete action and sends the database modification commands to the relational database storage device. Claim 46 recites determining one or more default delete actions based on the structure of the relational database. Claim 46 recites additional features that are not addressed in this rejection.

With regard to the remaining claims, these claims are distinguished over Ng and Sarkar for the reasons set forth in the response filed July 16, 2003 and for the reasons

noted above with regard to similar features found in claims 1, 12 and 46. In view of the above, Applicant again respectfully submits that neither Ng nor Sarkar, either alone or in combination, teach or suggest the features of independent claims 1, 12 and 46 or dependent claims 3-4, 7-11, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 47-48. Therefore, Applicant respectfully requests withdrawal of the rejection of claims 1, 3-4, 7-12, 14-15, 17-19, 25-26, 31-34, 39-42, 44 and 46-48 under 35 U.S.C. § 103(a).

II.   **35 U.S.C. § 103, Alleged Obviousness, Claims 5-6, 16, 22-23, 29-30, 37-38 and 45**

The Office Action rejects claims 5-6, 16, 22-23, 29-30, 37-38 and 45 under 35 U.S.C. § 103(a) as being unpatentable over Ng in view of Sarkar and further in view of Crus (U.S. Patent No. 4,947,320). Because this rejection is essentially the same as in the previous Office Action, this rejection is respectfully traversed for the reasons stated in the previous Response filed July 16, 2003, the remarks of which are hereby incorporated by reference. The following remarks are provided in rebuttal of the Examiner's statements in the present Office Action beginning on page 3, paragraph 2 of the Final Office Action.

In the July 16, 2003 Response, Applicant argued that neither Ng, Sarkar nor Crus teach that a delete action or the delete action identifier is one of cascade delete and nullify columns. In response, the Examiner, on page 4 of the present Office Action states the following:

> Ng does teach deletion or modifications the structure of a database via a programmed source code file. The deletion action of Ng does not clearly include the cascade deletion and nullify columns delete. But, Ng and Crus in combination teach the delete set null and delete cascade (col. 17, lines 7-29 and col. 18, line 1-26 and see the code in the table in col. 18). The Delete Set Null is set to nullify all the columns and the Delete Cascade is to delete all records in the dependent table or the structure of a table in the relational database (col. 17, line 17-28).

Applicant agrees with the Examiner that Ng does not teach a delete action or the delete action identifier is one of cascade delete and nullify columns. While Crus may teach a delete set null and a delete cascade operation in column 17, there is no teaching or suggestion to identify a delete set null or a delete cascade operation based on a

determined structure of a relational database, as determined from a meta-information class object associated with a relational database. Moreover, there is no teaching or suggestion in Crus to include a delete set null or a delete cascade operation identifier, for each dependent table of a plurality of tables in a relational database, in a meta-information class object. Therefore, there is no teaching or suggestion in Crus regarding a delete action (claims 6, 22-23, 29-30, 37-38 and 45) or delete action identifier (claims 5 and 16) being one of a cascade delete and nullify columns delete as recited in the above claims.

The Final Office Action's response to Applicant's arguments does not address these specific arguments but rather simply reiterates the rejection. The Final Office Action has failed to illustrate how Applicant's arguments are in error or why Applicant's arguments should not be considered persuasive. Therefore, Applicant respectfully submits that the arguments presented in the July 16, 2003 Response should be considered.

In view of the above, Applicant again respectfully submits that neither Ng, Sarkar nor Crus, either alone or in combination, teach or suggest the features of independent claims 1, 12, 20, 27, 35 and 43. At least by virtue of their dependency on claims 1, 12, 20, 27, 35 and 43, Ng, Sarkar and Crus, either alone or in combination, do not teach or suggest the features of dependent claims 5, 6, 16, 22-23, 29-30, 37-38 and 45. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 5, 6, 16, 22-23, 29-30, 37-38 and 45 under 35 U.S.C. § 103(a).

III.   35 U.S.C. § 103, Alleged Obviousness, Claims 20-21, 24, 27-28, 35-36, 43 and 46

The Office Action rejects claims 20-21, 24, 27-28, 35-36, 43 and 46 under 35 U.S.C. § 103(a) as unpatentable over Ng. Because this rejection is essentially the same as in the previous Office Action, this rejection is respectfully traversed for the reasons stated in the previous Response filed July 16, 2003, the remarks of which are hereby incorporated by reference. The following remarks are provided in rebuttal of the

Examiner's statements in the present Office Action beginning on page 3, paragraph 2 of the Final Office Action.

In the July 16, 2003 Response, Applicant argued that Ng does not teach or suggest determining a delete action based on a determined structure of a relational database. The Final Office Action fails to provide rebuttal on Applicant's remarks with respect to claims 20-21, 24, 27-28, 35-36, 43 and 46 from the Response dated July 16, 2003. Applicant holds that those remarks are still valid as earlier presented.
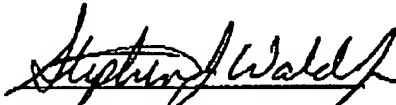
In view of the above, Applicant again respectfully submits that Ng does not teach or suggest the features of independent claims 20, 27, 35, 43 and 46. At least by virtue of their dependency on claims 20, 27 and 35, Ng does not teach or suggest the features of dependent claims 21, 24, 28 and 36. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 520-21, 24, 27-28, 35-36, 43 and 46 under 35 U.S.C. § 103(a).

IV.   Conclusion

It is respectfully urged that the subject application is patentable over Ng, Sarkar and Crus and is now in condition for allowance. The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

                                              Respectfully submitted,

DATE: _December 2, 2003_

                                              Stephen J. Walder, Jr.
                                              Reg. No. 41,534
                                              Carstens, Yee & Cahoon, LLP
                                              P.O. Box 802334
                                              Dallas, TX 75380
                                              (972) 367-2001
                                              Attorney for Applicant

SJW/fl

Page 20 of 20
George – 09/544,274